Exploratory Analysis of NOAA Storm Database

Synopsis

The document is about the exploratory analysis of the NOAA Storm Database. The dataset has been taken from this link. The dataset defines the enviornmental events, like typhoon, thunderstorms, floods, flash floods etc. which have occured since 1950 till 2011. The dataset also describes the economical loss that each of the events have caused to the country. Count for the number of injuries and fatalities that have occured due to each event has been defined too. The project is aimed at deriving the relationships and insights from the dataset about the injuries, property damage and the determining which event is the most dangerous event.

Loading and Preprocessing the data

We'll download the dataset from this link as a bz2 file. The format in which the data is available is a compressed form of the csv. This data is read using the read.csv function. The following code downloads the file from the internet and read it to create a dataframe.

```
download.file('https://d396qusza40orc.cloudfront.net/repdata%2Fdata%2FStormData.csv.bz2', 'StormData.csv
dataset = read.csv('StormData.csv.bz2')
```

The dimsensions of the dataset are as follows:

```
dim(dataset)
```

```
## [1] 902297     37
```

The contents along with the datatypes of the columns in the dataset are as follows:

```
str(dataset)
```

```
## 'data.frame':    902297 obs. of  37 variables:
##  $ STATE__   : num  1 1 1 1 1 1 1 1 1 1 ...
##  $ BGN_DATE  : Factor w/ 16335 levels "1/1/1966 0:00:00",..: 6523 6523 4242 11116 2224 2224 2260 383
##  $ BGN_TIME  : Factor w/ 3608 levels "00:00:00 AM",..: 272 287 2705 1683 2584 3186 242 1683 3186 318
##  $ TIME_ZONE : Factor w/ 22 levels "ADT","AKS","AST",..: 7 7 7 7 7 7 7 7 7 7 ...
##  $ COUNTY    : num  97 3 57 89 43 77 9 123 125 57 ...
##  $ COUNTYNAME: Factor w/ 29601 levels "","5NM E OF MACKINAC BRIDGE TO PRESQUE ISLE LT MI",..: 13513 1
##  $ STATE     : Factor w/ 72 levels "AK","AL","AM",..: 2 2 2 2 2 2 2 2 2 2 ...
##  $ EVTYPE    : Factor w/ 985 levels "   HIGH SURF ADVISORY",..: 834 834 834 834 834 834 834 834 834 8
##  $ BGN_RANGE : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ BGN_AZI   : Factor w/ 35 levels "","  N"," NW",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ BGN_LOCATI: Factor w/ 54429 levels "","- 1 N Albion",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ END_DATE  : Factor w/ 6663 levels "","1/1/1993 0:00:00",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ END_TIME  : Factor w/ 3647 levels "","  0900CST",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ COUNTY_END: num  0 0 0 0 0 0 0 0 0 0 ...
##  $ COUNTYENDN: logi  NA NA NA NA NA NA ...
##  $ END_RANGE : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ END_AZI   : Factor w/ 24 levels "","E","ENE","ESE",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ END_LOCATI: Factor w/ 34506 levels "","- .5 NNW",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ LENGTH    : num  14 2 0.1 0 0 1.5 1.5 0 3.3 2.3 ...
```

```
##  $ WIDTH     : num  100 150 123 100 150 177 33 33 100 100 ...
##  $ F         : int  3 2 2 2 2 2 2 1 3 3 ...
##  $ MAG       : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ FATALITIES: num  0 0 0 0 0 0 0 0 1 0 ...
##  $ INJURIES  : num  15 0 2 2 2 6 1 0 14 0 ...
##  $ PROPDMG   : num  25 2.5 25 2.5 2.5 2.5 2.5 2.5 25 25 ...
##  $ PROPDMGEXP: Factor w/ 19 levels "","-","?","+",..: 17 17 17 17 17 17 17 17 17 17 ...
##  $ CROPDMG   : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ CROPDMGEXP: Factor w/ 9 levels "","?","0","2",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ WFO       : Factor w/ 542 levels "","  CI","$AC",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ STATEOFFIC: Factor w/ 250 levels "","ALABAMA, Central",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ ZONENAMES : Factor w/ 25112 levels "","
##  $ LATITUDE  : num  3040 3042 3340 3458 3412 ...
##  $ LONGITUDE : num  8812 8755 8742 8626 8642 ...
##  $ LATITUDE_E: num  3051 0 0 0 0 ...
##  $ LONGITUDE_: num  8806 0 0 0 0 ...
##  $ REMARKS   : Factor w/ 436781 levels "","-2 at Deer Park\n",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ REFNUM    : num  1 2 3 4 5 6 7 8 9 10 ...
```

The libraries used in the project are mentioned below.

```
library(dplyr)
library(stringr)
library(gridExtra)
library(ggplot2)
library(usmap)
library(lubridate)
```

For the analysis of the dataset, we'll try to understand and clean the values present in PROPDMGEXP and EVTYPE columns of the dataset.

The unique values in the column are as follows:

```
unique(dataset$PROPDMGEXP)
```

```
##  [1] K M   B m + 0 5 6 ? 4 2 3 h 7 H - 1 8
## Levels:  - ? + 0 1 2 3 4 5 6 7 8 B h H K m M
```

We can map K to thousands, B to Billions, M to Millions and H to Hundreds. The data obtained from official NOAA website has been used to identify the meaning of variables in PROPDMGEXP column. Using the data from the website we can conclude that, the numerical correspond to 10 and '+', '-' correspond to 1 and '?', " correspond to 0.

Replacing the values we obtained from out analysis above into the dataset. Before doing that we'll have to change the casing of the characters in the column to upper case.

```
dataset$PROPDMGEXP = str_trim(dataset$PROPDMGEXP, side = 'both')
dataset$PROPDMGEXP = factor(dataset$PROPDMGEXP, levels = c('-', '?', '+', '0', '1', '2', '3', '4', '5',
dataset$PROPDMGEXP = as.numeric(as.character(dataset$PROPDMGEXP))
```

There are a lot of duplicates in the EVTYPE column. For transforming, we'll change the casing of the column to lower case and remove all the numerical values in the column.

```
dataset$EVTYPE = str_squish(str_to_lower(dataset$EVTYPE))
dataset$EVTYPE = str_trim(str_replace_all(dataset$EVTYPE, '\\(.*\\)|[.0-9]|\\)', ''))
```

The above code removed all the code which was written in a very detailed way like, thunderstorm wind (g24) to thunderstorm wind.

1. There are a lot of entries with 'tstm', we'll replace it with 'thunderstorm'.
2. There are a lot of duplicates like thunderstrom winds, thunderstorm wind. Replacing all winds with wind.
3. There are a lot of entries which define thunderstorm in a detailed way like thunderstorm wind / flood, replacing it with thunderstorm wind.
4. There are a lot of entries with floods defined in various tenses, replacing them with flood. For example, floods or flooding is replaced with flood
5. Blizzard has been defined in various ways. We'll reconcile it to be represented in a single format as blizzard.
6. The weather related to snow has been defined in various ways. Replacing it with Winter Storm.
7. The weather about cold has been defined in various ways, replacing it with winter weather.
8. Simlarly repalcing the warm, heat and hot realted words with heat.
9. Flood has been defined in various ways. Replacing each with flood.
10. Flood has been defined as fld. Replacing it with flood
11. Replacing to flood only if flash is not present in the string.
12. Hurricanes have been named along with thier names. Replacing them with just hurricane.
13. Replacing values with dry with drought
14. Replacing values with hail in it with hail.
15. Replacing all the words with freeze or frost with freeze/frost
16. Replacing all the surf realted strings to high surf
17. Tropical storms have been named. Updating ti to just tropical storm
18. Replacing all the fire related strings to wildfire.
19. wet signifies rain. So replacing the values with heavy rain
20. Replacing all the frost related strings to freeze/frost
21. Changing the string combination of waterspount to waterspount
22. Replacing all the strings with lightning in it with lightning
23. Replacing high wind and tide related values to storm surge/wind
24. Replacing all the strings containing tornado to tornado
25. Replacing showers with heavy rain.

```
a = str_replace_all(dataset$EVTYPE, 'tstm', 'thunderstorm')
a = str_replace_all(a, 'winds', 'wind')
a = str_replace_all(a, '.*thunderstorm.*', 'thunderstorm wind')
a = str_replace_all(a, 'flood(.*)', 'flood')
a = str_replace_all(a, '.*blizzard.*', 'blizzard')
a = str_replace_all(a, '.*snow.*', 'winter storm')
a = str_replace_all(a, '.*cold.*', 'winter weather')
a = str_replace_all(a, '.*warm.*', 'heat')
a = str_replace_all(a, '.*heat.*', 'heat')
a = str_replace_all(a, '.*hot.*', 'heat')
a = str_replace_all(a, '.*rain.*', 'heavy rain')
a = str_replace_all(a, '.*ice.*', 'ice storm')
a = str_replace_all(a, ' fld.*', ' flood')
a = str_replace_all(a, '.*^(?!flash).*flood.*', 'flood')
a = str_replace_all(a, '.*hurricane.*', 'hurricane')
a = str_replace_all(a, '.*dry.*', 'drought')
a = str_replace_all(a, '.*hail.*', 'hail')
```

```
a = str_replace_all(a, '.*surf.*', 'high surf')
a = str_replace_all(a, '.*tropical storm.*', 'tropical storm')
a = str_replace_all(a, '.*fire.*', 'wildfire')
a = str_replace_all(a, '.*wet.*', 'heavy rain')
a = str_replace_all(a, '.*frost.*', 'freeze/frost')
a = str_replace_all(a, '.*waterspout.*', 'waterspout')
a = str_replace_all(a, '.*lightning.*', 'lightning')
a = str_replace_all(a, '.*high.*tide.*', 'storm surge/tide')
a = str_replace_all(a, '.*tornado.*', 'tornado')
a = str_replace_all(a, '.*shower.*', 'heavy rain')
a = str_replace_all(a, '.*high wind.*', 'high wind')
dataset$EVTYPE = a
```

The EVTYPE column of the dataset is now relatively clean. Few spelling mistakes do exist, but the count of the erronious records is very low.

We'll identify the most frequent events that have occured during the time-span.

```
events <- dataset %>%
                group_by(EVTYPE) %>%
                filter(EVTYPE != '?') %>%
                summarise(No.Of.Occurances = n(),
                          Total_damage = sum(PROPDMG*PROPDMGEXP, na.rm = TRUE),
                          Total_injuries = sum(INJURIES, na.rm = TRUE),
                          Total_fatalities = sum(FATALITIES, na.rm = TRUE))
```

The contents of the dataset are as follows:

```
head(events)
```

```
## # A tibble: 6 x 5
##   EVTYPE      No.Of.Occurances Total_damage Total_injuries Total_fatalities
##   <chr>                  <int>        <dbl>          <dbl>            <dbl>
## 1 agricultur~                6            0              0                0
## 2 apache cou~                1         5000              0                0
## 3 astronomic~              174       320000              0                0
## 4 avalance                   1            0              0                1
## 5 avalanche                386      3721800            170              224
## 6 beach eros~                1            0              0                0
```

Rearranging the dataset according to number of incidents

```
events = events[order(events$No.Of.Occurances, decreasing = TRUE), ]
```

Most Frequent Events

The following dataframe describes the top 10 most frequent events.

```
head(select(events, EVTYPE, No.Of.Occurances), 10)
```

```
## # A tibble: 10 x 2
##    EVTYPE            No.Of.Occurances
##    <chr>                        <int>
##  1 thunderstorm wind           336808
##  2 hail                        289282
##  3 tornado                      60684
##  4 flash flood                  55037
##  5 flood                        31056
##  6 winter storm                 29131
##  7 high wind                    21915
##  8 lightning                    15760
##  9 heavy rain                   12225
## 10 winter weather                9499
```

Rearranging the dataset according to

```
events = events[order(events$Total_damage, decreasing = TRUE), ]
```

Events with the most destruction

The dataframe given below describes the top 10 events which have caused the most financial losses defined in US dollars($).

```
head(select(events, EVTYPE, Total_damage), 10)
```

```
## # A tibble: 10 x 2
##    EVTYPE            Total_damage
##    <chr>                    <dbl>
##  1 flood              150837629534
##  2 hurricane           84756180010
##  3 tornado             56941934097
##  4 storm surge         43323536000
##  5 flash flood         16732372111
##  6 hail                15974566877
##  7 thunderstorm wind   12576178222
##  8 wildfire             8501628500
##  9 winter storm         7716297017
## 10 tropical storm       7714390550
```

Injuries and Fatalities caused by the Events

Here we'll plot a subplot, one for the top 10 events with the most injuries and fatalities.

Rearranging the Total_injuries column in a descending manner.

```
events = events[order(events$Total_injuries, decreasing = TRUE), ]
```

```
head(select(events, EVTYPE, Total_injuries), 10)
```

```
## # A tibble: 10 x 2
```

```
##    EVTYPE            Total_injuries
##    <chr>                      <dbl>
##  1 tornado                    91364
##  2 thunderstorm wind           9544
##  3 heat                        9243
##  4 flood                       6896
##  5 lightning                   5231
##  6 winter storm                2486
##  7 ice storm                   2154
##  8 flash flood                 1785
##  9 wildfire                    1608
## 10 high wind                   1476
```

Rearranging the column Total_fatalities in descending manner

```
events = events[order(events$Total_fatalities, decreasing = TRUE), ]
```

```
head(select(events, EVTYPE, Total_fatalities), 10)
```

```
## # A tibble: 10 x 2
##    EVTYPE            Total_fatalities
##    <chr>                        <dbl>
##  1 tornado                       5633
##  2 heat                          3178
##  3 flash flood                   1018
##  4 lightning                      817
##  5 thunderstorm wind              754
##  6 flood                          535
##  7 winter weather                 469
##  8 winter storm                   375
##  9 rip current                    368
## 10 high wind                      292
```

State wise analysis of the events

We have established the most frequent events, the ones which cause most financial damage, and the ones which cause the most injuries and fatalities.

Now we'll try to analyse the geographical regions/state which have had the most number of calamities. In order to eastablish this, we'll group the dataset on the basis of states and identify the count for each state.

```
states = dataset %>%
            group_by(STATE) %>%
            summarise(Count = n())
```

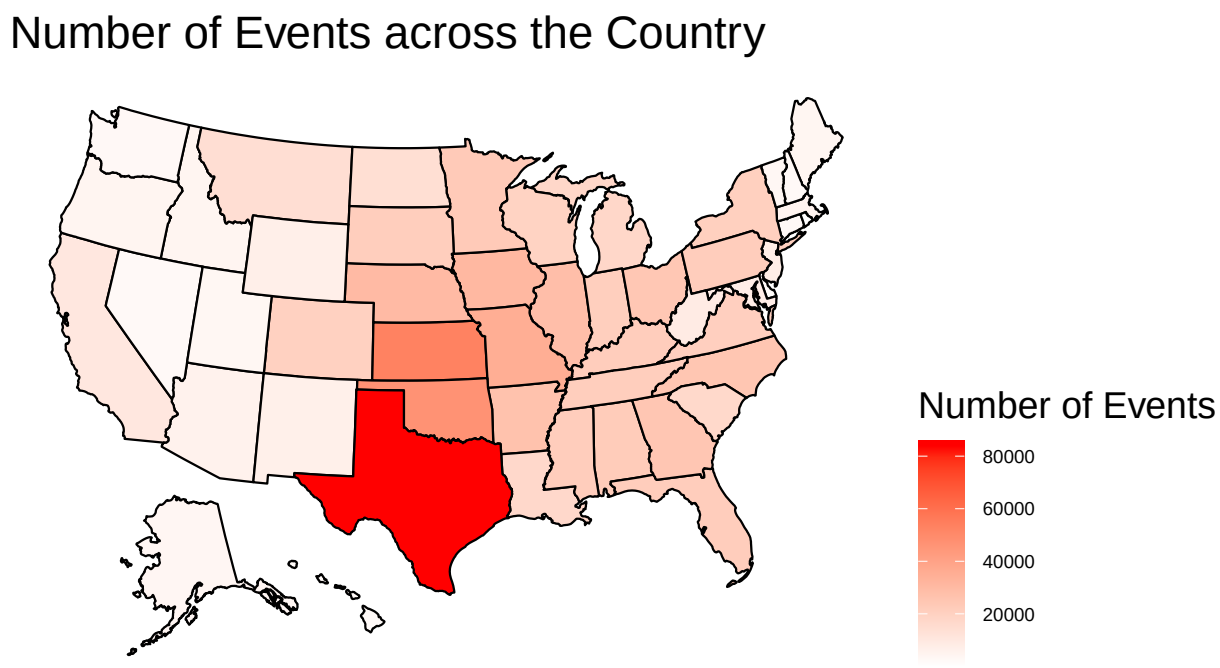
Arranging the Count in descending order,

```
states = states[order(states$Count, decreasing = TRUE), ]
colnames(states) <- c('state', 'count')
head(states)
```

```
## # A tibble: 6 x 2
##   state count
##   <fct> <int>
## 1 TX    83728
## 2 KS    53440
## 3 OK    46802
## 4 MO    35648
## 5 IA    31069
## 6 NE    30271
```

As we can see above, Texas runs ahead with the majority of the events. All the states Texas, Kansas, Oklahoma, Missouri, Indiana and Nebraska, are all located in the central part of the country. We'll analyse which event has occurred the most in these states. The plot below displays the states according to the number of events.

```
plot_usmap(data = states, value = 'count') +
    scale_fill_continuous(low = 'white', high = 'red', name = 'Number of Events') +
    theme(legend.position = 'right', title = element_text(size = 14, hjust = 0.5)) +
    ggtitle('Number of Events across the Country')
```




```
state_events = dataset %>%
                    filter(STATE %in% c('TX', 'KS', 'OK', 'MO', 'IA', 'NE')) %>%
                    group_by(STATE, EVTYPE) %>%
                    summarise(Count = n())
state_events = state_events[order(state_events$Count, decreasing = T), ]
head(state_events, 10)
```

```
## # A tibble: 10 x 3
## # Groups:   STATE [6]
##    STATE EVTYPE            Count
##    <fct> <chr>             <int>
##  1 TX    hail              37209
##  2 KS    hail              28470
##  3 OK    hail              23701
##  4 TX    thunderstorm wind 23394
##  5 NE    hail              16357
##  6 KS    thunderstorm wind 15546
##  7 OK    thunderstorm wind 14939
##  8 MO    hail              14193
##  9 MO    thunderstorm wind 12186
## 10 IA    thunderstorm wind 11295
```

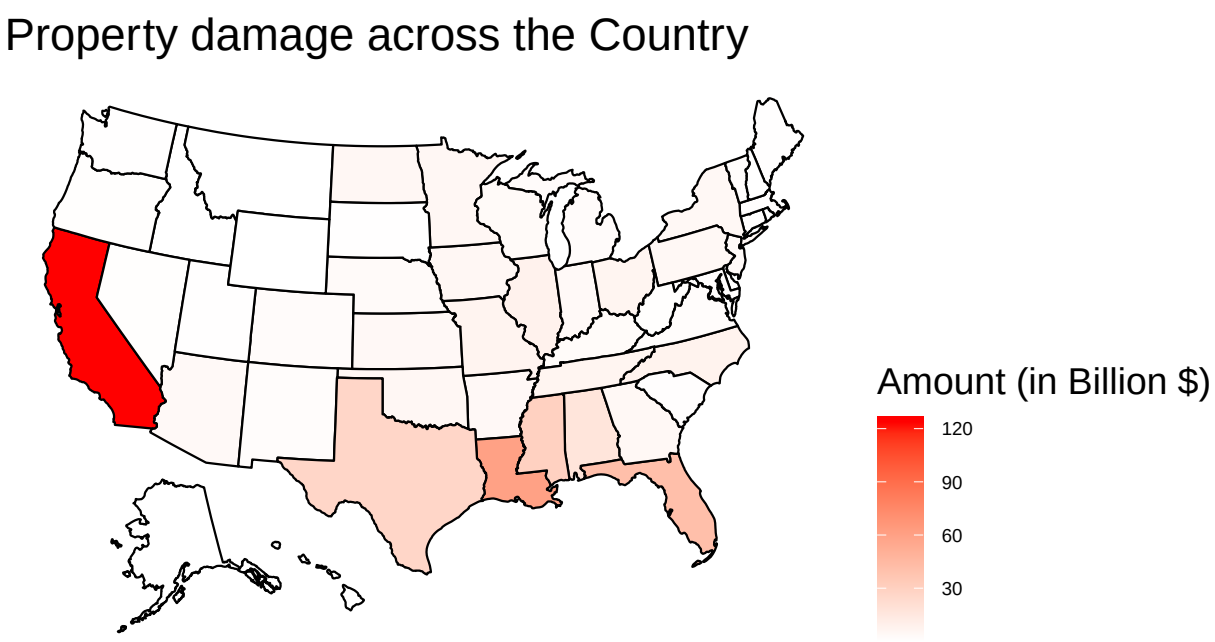
We can see that hail is the most occured event for each state. We can also determine that hail and thunderstorm account for more than 50% of the events for each state.

Statewise property damage

For calculating the property damage, we'll have to group the data on the basis of states and calculate the amount of of damage by multiplying the PROPDMG and PROPDMGEXP columns.

```
prop_dmg = dataset %>%
                group_by(STATE) %>%
                summarise(Property_damage = sum(PROPDMG*PROPDMGEXP/1000000000,
                                                na.rm = TRUE))
colnames(prop_dmg) = c('state', 'property_damage')

plot_usmap(data = prop_dmg, value = 'property_damage') +
    scale_fill_continuous(low = 'white', high = 'red',
                          name = 'Amount (in Billion $)') +
    theme(legend.position = 'right', title = element_text(size = 14, hjust = 0.5)) +
    ggtitle('Property damage across the Country')
```



We can see from the plot that California has experienced the most damage due to events in the entire country.

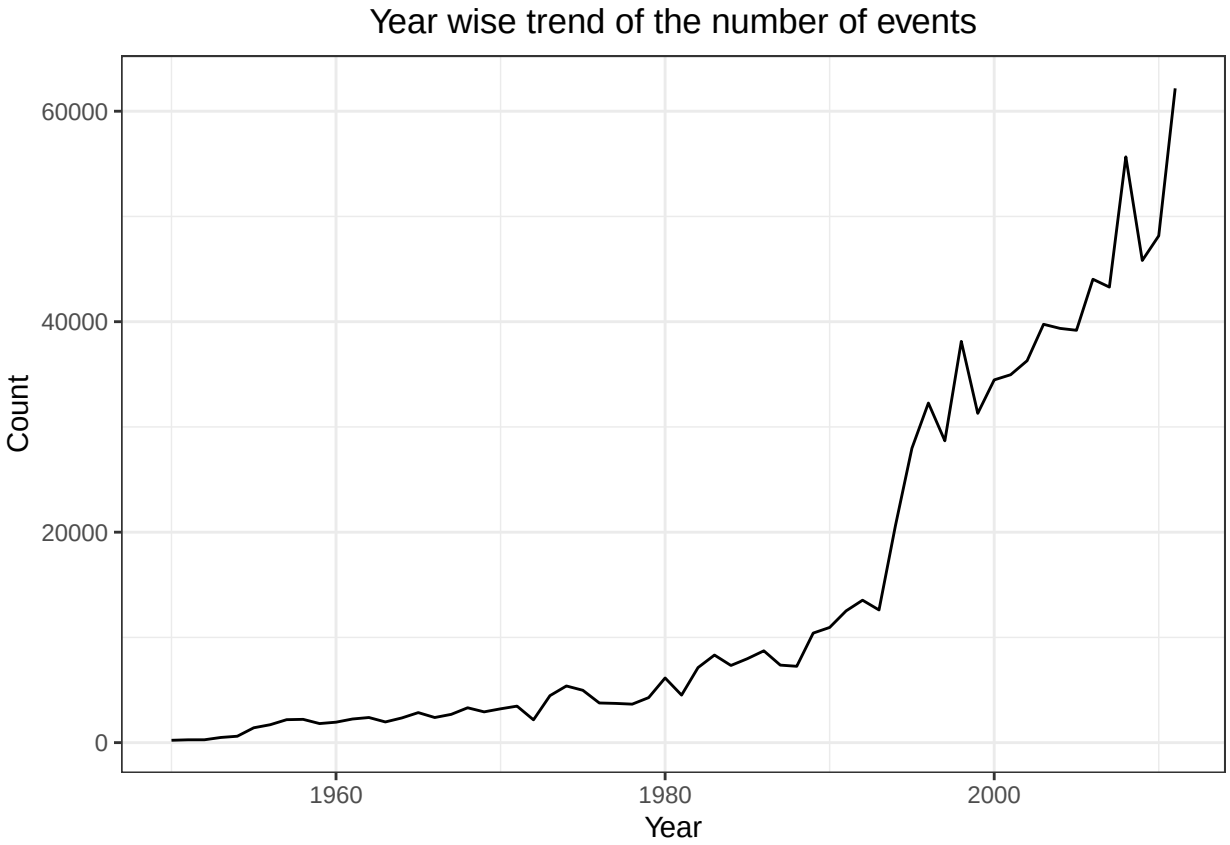
#Trend of the events over the years

For determining the trend of number of events across the years, we'll add a new column to the dataset which will have the year of the event.

```
dataset$year = year(mdy_hms(dataset$BGN_DATE))

year_events = dataset %>%
                    group_by(year) %>%
                    summarise(Count = n())

ggplot(year_events) +
    geom_line(aes(year, Count)) +
    theme_bw()+
    labs(title = 'Year wise trend of the number of events', x = 'Year', y = 'Count')     +
    theme(plot.title = element_text(hjust = 0.5))
```



Results

Following are the observations from the storm dataset:

1. Thunderstorm and hail are the most frequent type of events across the country.
2. Flood and Hurricane are the events which cause the most property damage.
3. Tornado tops the list for the number of injuries and fatalities caused due to an event.
4. Heat which was not even in the top 10 events of the country, comes second to tornado as the event with most number of fatalities.
5. Statewise analysis highlights the states with the most number of events. Texas, Kansas, Oklahoma, Missouri, Indiana and Nebraska have the most number of incidents.
6. California which had relatively less count for number of events, tops the list for the property damage at more than $120 billion.
7. The number of events has seen a rising trend since the dataset has been maintained.